

Lab 1 - Cloud Computing

Okiki Ojo (100790236)

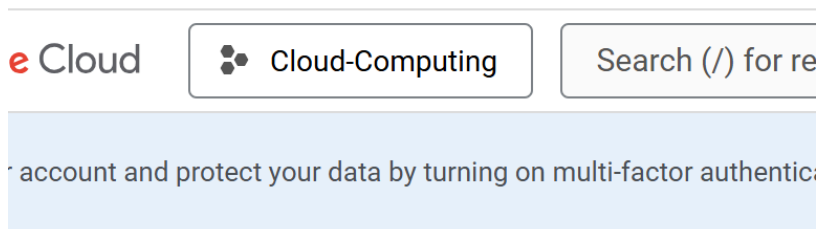
GitHub Link: <https://github.com/okikio-school/cloud-computing-labs/tree/main/Lab%201>

Video Links:

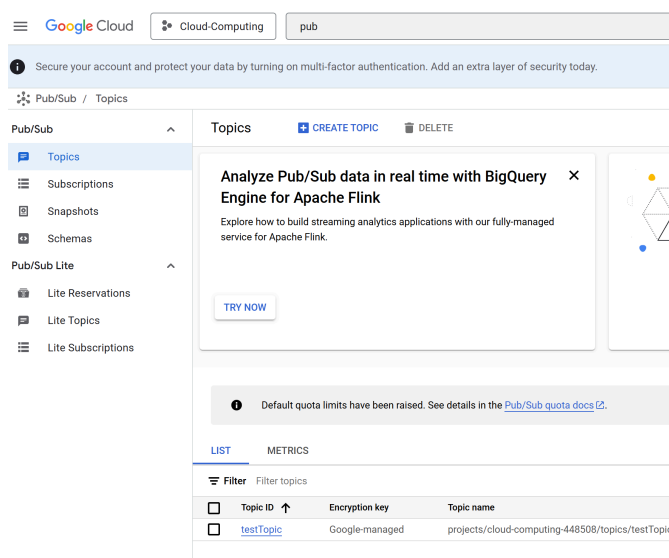
- **Smart Meter Video:** 📺 SmartMeter - Screen Recording 2025-01-23 011927.mp4
- **Design Part Video:** 📺 Design Part - Screen Recording 2025-01-23 020435.mp4

Activity

Step 1. Create GCP Project



Step 2. Create Pub/Sub called testTopic



Step 3. Create Service Account called pubsub-system

←

Create service account

1

Service account details

Service account name

pubsub-system

Display name for this service account

Service account ID *

pubsub-system-703

Email address: pubsub-system-703@cloud-computing-448508.iam.gserviceaccount.com

Service account description

Describe what this service account will do

CREATE AND CONTINUE

Step 4. Create Keys for pubsub-system

←

pubsub-system

DETAILS

PERMISSIONS

KEYS

METRICS

LOGS

Keys

⚠

Service account keys could pose a security risk if compromised. We recommend you avoid downloading Federation. Learn more about the best way to authenticate service accounts on Google Cloud.

ℹ

Google automatically disables service account keys detected in public repositories. You can customize the 'iam.serviceAccountKeyExposureResponse' organization policy. Learn more

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organization policies. Learn more about setting organization policies for service accounts

ADD KEY

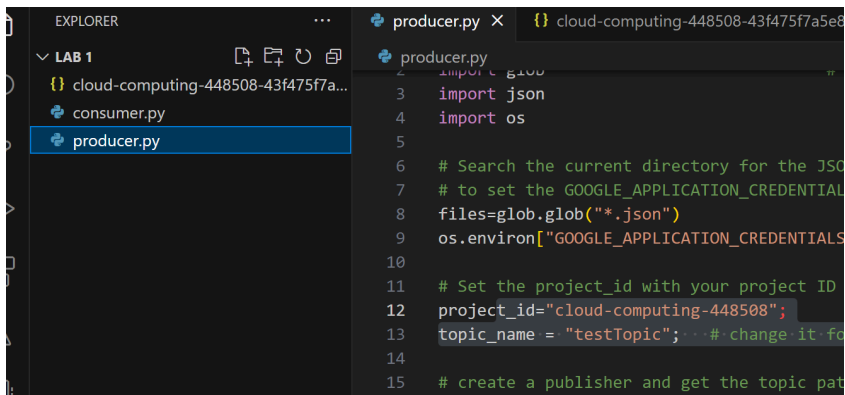
Type	Status	Key	Creation date	Expiration date
	Active	43f475f7a5e81fc1b3e41a9c127e1f913caa1eab	Jan 22, 2025	Dec 31, 9999

Generated key (Google provided)

Step 5. Configure the Project ID from the Cloud Project ID

```
5
6 # Search the current directory for the JSON file (including the service
7 # to set the GOOGLE_APPLICATION_CREDENTIALS environment variable.
8 files=glob.glob("*.json")
9 os.environ["GOOGLE_APPLICATION_CREDENTIALS"]=files[0];
10
11 # Set the project_id with your project ID
12 project_id="cloud-computing-448508";
13 topic_name = "testTopic"; # change it for your topic name if needed
14
15 # create a publisher and get the topic path for the publisher
16 publisher = pubsub_v1.PublisherClient()
17 topic_path = publisher.topic_path(project_id, topic_name)
18 print(f"Published messages with ordering keys to {topic_path}.")
19
20 # Iterate for 100 times
```

Step 6. Downloaded service account keys as JSON



```
1 import glob
2
3 import json
4 import os
5
6 # Search the current directory for the JSON
7 # to set the GOOGLE_APPLICATION_CREDENTIAL
8 files=glob.glob("*.json")
9 os.environ["GOOGLE_APPLICATION_CREDENTIALS"]
10
11 # Set the project_id with your project ID
12 project_id="cloud-computing-448508";
13 topic_name = "testTopic"; # change it fo
14
15 # create a publisher and get the topic pat
```

Step 7. Install the python google-cloud-pubsub library into Anaconda

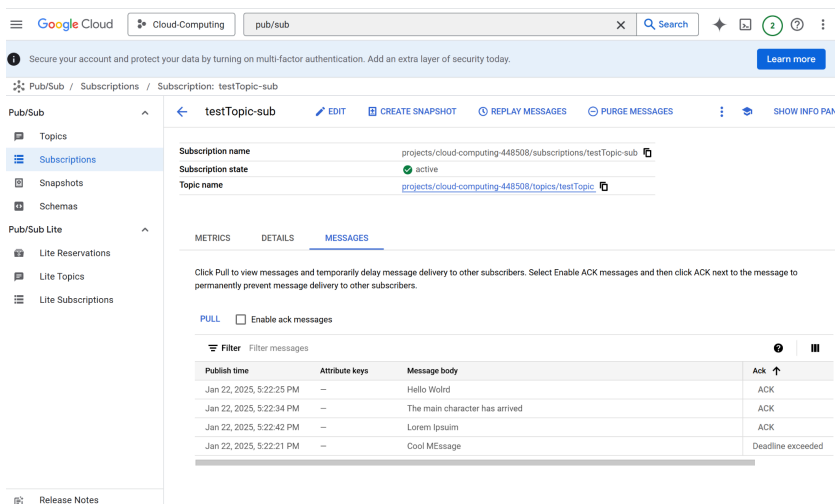
```
(base) C:\Users\femik>pip install google-cloud-pubsub
Collecting google-cloud-pubsub
  Downloading google_cloud_pubsub-2.27.2-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting grpcio<2.0dev,>=1.51.3 (from google-cloud-pubsub)
  Downloading grpcio-1.69.0-cp312-cp312-win_amd64.whl.metadata (4.0 kB)
Collecting google-auth<3.0.0dev,>=2.14.1 (from google-cloud-pubsub)
  Downloading google_auth-2.37.0-py2.py3-none-any.whl.metadata (4.8 kB)
Collecting google-api-core!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0dev,>=1.34.0 (from google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0dev,>=1.34.0->google-cloud-pubsub)
  Downloading google_api_core-2.24.0-py3-none-any.whl.metadata (3.0 kB)
Collecting proto-plus<2.0.0dev,>=1.22.0 (from google-cloud-pubsub)
  Downloading proto_plus-1.25.0-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4
```

Step 8. Run the producer.py file and enter messages to produce to the pub/sub

```
(base) C:\Users\femik>cd C:\Users\femik\OneDrive - Ontario Tech University\Year 4\Cloud Computing\Lab 1

(base) C:\Users\femik\OneDrive - Ontario Tech University\Year 4\Cloud Computing\Lab 1>python producer.py
Published messages with ordering keys to projects/cloud-computing-448508/topics/testTopic.
Enter a value (String):Cool MESSagE
Producing a record: b'Cool MESSagE '
Enter a value (String):Hello Wo!rd
Producing a record: b'Hello Wo!rd'
Enter a value (String):The main character has arrived
Producing a record: b'The main character has arrived'
Enter a value (String):Lorem Ipsuim
Producing a record: b'Lorem Ipsuim'
Enter a value (String):
```

Step 9. Pull the messages for viewing



Google Cloud Cloud-Computing pub/sub

Secure your account and protect your data by turning on multi-factor authentication. Add an extra layer of security today. [Learn more](#)

Pub/Sub / Subscriptions / Subscription: testTopic-sub

Pub/Sub

- Topics
- Subscriptions**
- Snapshots
- Schemas

Pub/Sub Lite

- Lite Reservations
- Lite Topics
- Lite Subscriptions

testTopic-sub

EDIT CREATE SNAPSHOT REPLAY MESSAGES PURGE MESSAGES SHOW INFO PAN

Subscription name projects/cloud-computing-448508/subscriptions/testTopic-sub

Subscription state active

Topic name projects/cloud-computing-448508/topics/testTopic

METRICS DETAILS **MESSAGES**

Click Pull to view messages and temporarily delay message delivery to other subscribers. Select Enable ACK messages and then click ACK next to the message to permanently prevent message delivery to other subscribers.

PULL ☐ Enable ack messages

Filter Filter messages

Publish time	Attribute keys	Message body	Ack
Jan 22, 2025, 5:22:25 PM	—	Hello Wo!rd	ACK
Jan 22, 2025, 5:22:34 PM	—	The main character has arrived	ACK
Jan 22, 2025, 5:22:42 PM	—	Lorem Ipsuim	ACK
Jan 22, 2025, 5:22:21 PM	—	Cool MESSagE	Deadline exceeded

Release Notes

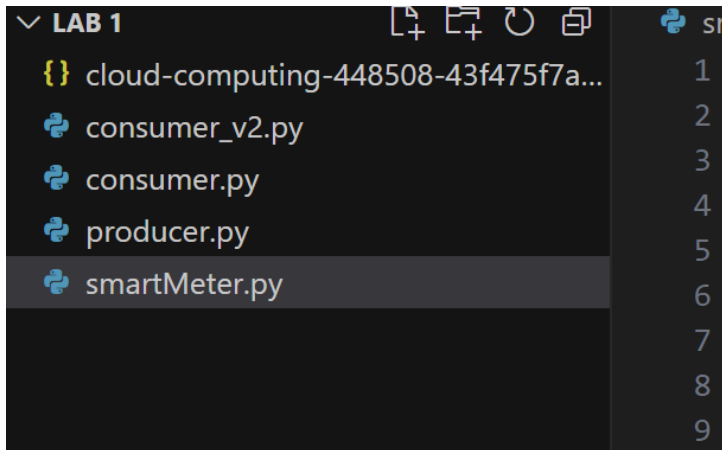
Step 10. Running the consumer.py file to pull messages from the pub/sub

```
(base) C:\Users\femik>cd C:\Users\femik\OneDrive - Ontario Tech University\Year 4\Cloud Computing\Lab 1

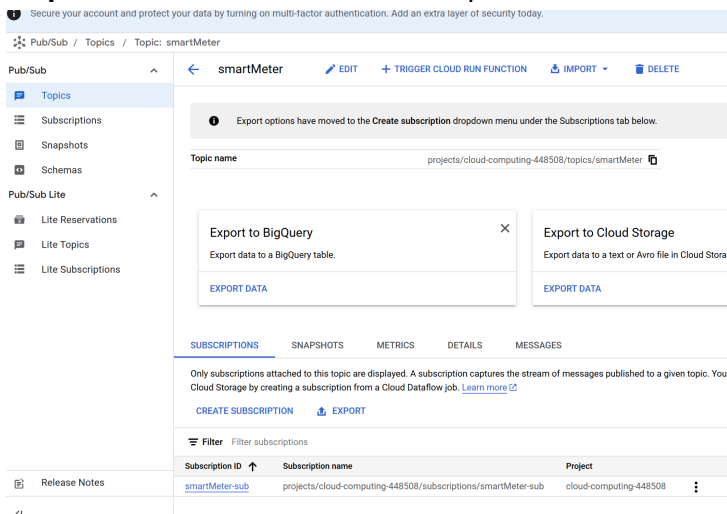
(base) C:\Users\femik\OneDrive - Ontario Tech University\Year 4\Cloud Computing\Lab 1>python consumer.py
Listening for messages on projects/cloud-computing-448508/subscriptions/testTopic-sub..

Consumed record with value : b'Cool MESSagE '
Consumed record with value : b'Lorem Ipsuim'
Consumed record with value : b'The main character has arrived'
Consumed record with value : b'Hello Wo!rd'
```

Step 11. Copy smartMeter.py & consumer_v2.py (I later placed both smartMeter & consumer into the ./v2 directory so the github code isn't a 1-to-1 match)



Step 12. Create the Smart Meter Topic



Discussion

Question 1: What is EDA? What are its advantages and disadvantages?

EDA = Event Driven Architecture; it's a design pattern that is primarily driven by events, such that the architecture supports completely isolated and independent event producers and consumers, allowing each to independently act in response to an event and/or to send an event.

- **Pros:**
 - Scalable
 - Performant
 - Loose coupling

- Highly flexible
- Supports Parallel Processing
- Supports Asynchronous Message Passing
- Fault tolerant/Resilient
- Supports Real-time processing
- **Cons:**
 - Adds Complexity
 - Event Ordering & Consistency can be issues
 - Dependent on the Event Broker as single points of failure

Question 2: Cloud Pub/Sub has two types of subscriptions: push and pull. Describe them, showing the strengths and weaknesses of each based on potential applications.

- **Push Subscriptions:**
 - Have the Pub/Sub topic continuously push messages when they are ready.
 - The only issue is if the subscriber doesn't acknowledge the event the Pub/Sub topic will have to continuously retry while trying to push the event to the subscriber, if after a number of retry attempts the subscriber doesn't acknowledge the event it gets discarded.
 - **Strengths:**
 - Low latency & great for Real-time processing e.g. great for real-time analytics
 - No need for manually polling on the subscriber; can have simpler clients, e.g. great for serverless functions
 - Simpler subscriber implementation as it just uses HTTPs endpoints to push the events to the subscriber e.g. since it's a simple HTTP endpoint they work really well for webhooks
 - **Weakness:**
 - Subscribers have less control over the event throughput, so lots of events from the Pub/Sub topic can easily overwhelm the subscriber; requires rate limiting or queueing to be setup e.g. bad for large batch data processing

- Requires high network reliability of the subscriber otherwise events that can't be delivered will be discarded by the Pub/Sub topic e.g. great for streaming video calls
- **Pull Subscriptions:**
 - Let the Pub/Sub topic know when the subscriber is ready for messages
 - It requires the subscriber to be up and continuously polling to request new messages
 - **Strengths:**
 - Subscriber is fully in control over when events arrive, it's great for clients which can't handle being flooded with events, e.g. a small mobile devices with spotty network connectivity
 - Resilient to Temporary Network issues; since subscribers only pull once they are ready, so whether there is a temporary disconnection or not it likely won't affect the subscriber e.g. great for batch processing
 - **Weakness:**
 - More load on the subscriber to continuously check for new events; not great when there is a low amount of events e.g. bad for real-time tasks like analytics
 - Higher latency; since events only arrive after requesting them there is a potential slowdown e.g. bad for streaming video calls

Question 3: When publishing a message into a topic, an ordering key can be specified. Using examples, describe the role and benefits of ordering keys.

- Ordering keys are a feature to ensure events published with the same key are delivered to subscribers in the same order every single time
- They can assist Cloud Pub/Sub in identifying the priority order to choose when events arrive at the same time
- Due to parallel processing nature of Pub/Sub the order of events can get all twisted, ordering keys can assist in ensuring a strict event sequence is kept
- **E.g. Processing a checkout:**

- A customer removes an item from their cart
- Then finalizes their purchase
- If the order is the opposite the customer will be very annoyed, ordering keys can always set it up such that all purchase events should be last after all other shopping events occur in time
- **E.g. Chat message:**
 - A customer sends a chat to a friend
 - If in that time the friend responds with their own message
 - If the order is wrong the chat history will be confusing
 - Using ordering keys you can ensure messages use the timestamp of the message and time it took the message sender to press and release the send button for determining which message should come first in the chat history

Smart Meter Screenshot:

The screenshot shows an Anaconda Prompt window with two panes. The left pane displays a series of log messages, each starting with 'Consumed record with value :'. The right pane shows a code editor with a file named 'smartMeter.py'.

Log Output (Left Pane):

```
Consumed record with value : {'time': 1737612692.18728, 'profile_name': 'denver', 'temperature': 31.336934595226534, 'humidity': 40.61861174886836, 'pressure': 1.0858354864089734}
Consumed record with value : {'time': 1737612692.711142, 'profile_name': 'denver', 'temperature': None, 'humidity': 46.887950338965325, 'pressure': None}
Consumed record with value : {'time': 1737612693.2407978, 'profile_name': 'losang', 'temperature': 71.46191382162336, 'humidity': None, 'pressure': 1.000601841187211}
Consumed record with value : {'time': 1737612693.772515, 'profile_name': 'boston', 'temperature': 32.124575571843664, 'humidity': 81.93609622921693, 'pressure': 0.992086279728241}
Consumed record with value : {'time': 1737612694.3056908, 'profile_name': 'denver', 'temperature': 52.53557750151686, 'humidity': 46.72932196980768, 'pressure': 2.04183692653509}
Consumed record with value : {'time': 1737612694.8351777, 'profile_name': 'boston', 'temperature': 40.15766187933961, 'humidity': 47.35266779954817, 'pressure': 0.9816704111473463}
Consumed record with value : {'time': 1737612695.3649235, 'profile_name': 'denver', 'temperature': None, 'humidity': 43.50997535952715, 'pressure': None}
Consumed record with value : {'time': 1737612695.8987367, 'profile_name': 'boston', 'temperature': 49.18397990352401, 'humidity': 87.12375022367178, 'pressure': 1.1317769893029577}
Consumed record with value : {'time': 1737612696.4274058, 'profile_name': 'denver', 'temperature': None, 'humidity': 24.669719656984547, 'pressure': 1.6067775676318077}
```

Code Editor (Right Pane):

```
1 from google.cloud import pubsub_v1
2 import glob
3 import json
4 import os
5 import random
6 import numpy as np
7 import time
8
9 # Search the current directory for the JSON files
10 # to set the GOOGLE_APPLICATION_CREDENTIALS
11 files=glob.glob("*.json")
12 os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = files[0]
13
14 # Set the project_id with your project ID
15 project_id="cloud-computing-448508";
16 topic_name = "smartMeter"; # change it
17
18 # create a publisher and get the topic path
19 publisher = pubsub_v1.PublisherClient()
```

Design Part Screenshot of Google Cloud (I forgot to record this as part of the video):



Google Cloud



Cloud-Computing

Search (/) for resources, docs, products, and more



Search



Pub/Sub / Topics / Topic: carDataTopic

Pub/Sub



Topics



Subscriptions



Snapshots



Schemas

Pub/Sub Lite



Lite Reservations



Lite Topics



Lite Subscriptions



carDataTopic



EDIT



TRIGGER CLOUD RUN FUNCTION



IMPORT



DELETE



Export options have moved to the **Create subscription** dropdown menu under the Subscriptions tab below.

Topic name

projects/cloud-computing-448508/topics/carDataTopic



SUBSCRIPTIONS

SNAPSHOTS

METRICS

DETAILS

MESSAGES

Only subscriptions attached to this topic are displayed. A subscription captures the stream of messages published to a given topic. You can store messages in Cloud Storage by creating a subscription from a Cloud Dataflow job. [Learn more](#)

[CREATE SUBSCRIPTION](#)



EXPORT



Filter Filter subscriptions

Subscription ID



Subscription name

Project

[carDataTopic-sub](#)

projects/cloud-computing-448508/subscriptions/carDataTopic-sub

cloud-computing-448508

